

## Enviando e recebendo mensagens JMS

### Downloads

Caso queira começar o treinamento a partir dessa aula, pode baixar o projeto [aqui][1]. Baixe este arquivo, se não tiver feito os exercícios anteriormente.

### Revisão do capítulo anterior

Falamos sobre o tratamento de erro. Vimos como configurar o `errorHandler`, usando um `deadLetterChannel` com `redelivery`. Mencionamos que o `deadLetterChannel` é uma adaptação de um conceito já usado na mensageria, por exemplo, JMS. Neste capítulo, vamos consumir e enviar mensagens de uma fila JMS.

### Instalação do ActiveMQ

Até agora, lemos as nossas mensagens de uma pasta. Isso pode acontecer na integração (seguindo o estilo de integração *File Transfer* ou *Troca de arquivo*), mas no caso, todas as mensagens vêm de um fila JMS. Para simular a entrega de mensagens, usaremos um MOM (*Message Oriented Middleware*) muito popular no mundo Java: o ActiveMQ.

Para instalar basta baixar o ZIP da página do ActiveMQ: [\(http://activemq.apache.org/download.html\)](http://activemq.apache.org/download.html)

Depois ter descompactado podemos entrar na pasta de instalação e na pasta `bin` (na linha de comando). Para subir o ActiveMQ basta rodar o script `activemq` com o parâmetro `console`.

- No Windows: `activemq.bat console`
- No Linux ou Mac: `sh activemq console`

Isso faz com que ActiveMQ suba mostrando informações no console.

Obs: O Alura possui um treinamento focado no JMS com ActiveMQ que vai muito além do apresentado aqui:  
[\(https://www.alura.com.br/course/jms\)](https://www.alura.com.br/course/jms)

### Preparação da fila pedidos

Agora já podemos acessar a interface de administração do ActiveMQ: <http://localhost:8161/admin/>

Para os dados de Login e Senha use **admin**.

Após fazermos o login, cadastraremos uma nova fila. Clique no *Queues* e chame a fila de pedidos :

Uma vez criada, enviaremos mensagens pela interface web. Clique no link "Send" da coluna "Operations":

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
pedidos	0	0	0	0	BROWSE ACTIVE CONSUMERS ACTIVE PRODUCERS 	Send To Purge Delete

Em seguida, veremos um formulário para enviar uma mensagem JMS. No corpo da mensagem, vamos colar o XML de um pedido:

### Send a JMS Message

<b>Message Header</b>			
Destination	pedidos	Queue or Topic	Queue
Correlation ID	<input type="text"/>	Persistent Delivery	<input type="checkbox"/>
Reply To	<input type="text"/>	Priority	<input type="text"/>
Type	<input type="text"/>	Time to live	<input type="text"/>
Message Group	<input type="text"/>	Message Group Sequence Number	<input type="text"/>
delay(ms)	<input type="text"/>	Time(ms) to wait before scheduling again	<input type="text"/>
Number of repeats	<input type="text"/>	Use a CRON string for scheduling	<input type="text"/>
Number of messages to send	<input type="text"/> 1	Header to store the counter	JMSXMessageCounter
<input type="button" value="Send"/> <input type="button" value="Redefinir"/>			
<b>Message body</b>		<pre>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt; &lt;pedido&gt;   &lt;id&gt;2451256&lt;/id&gt;   &lt;dataCompra&gt;2013-12-05T18:21:07.529-02:00&lt;/dataCompra&gt;   &lt;itens&gt;     &lt;item&gt;       &lt;formato&gt;EBOOK&lt;/formato&gt;       &lt;quantidade&gt;1&lt;/quantidade&gt;       &lt;livro&gt;         &lt;codigo&gt;ARQ&lt;/codigo&gt;         &lt;titulo&gt;Introdução à Arquitetura e Design de Software&lt;/titulo&gt;         &lt;tituloCurto&gt;Arquitetura Java&lt;/tituloCurto&gt;         &lt;nomeAutor&gt;Sergio Lopes, Paulo Silveira, Guilherme Silveira, Nico Steppat, outros&lt;/nomeAutor&gt;         &lt;valorEbook&gt;29.90&lt;/valorEbook&gt;         &lt;valorImpresso&gt;79.90&lt;/valorImpresso&gt;       &lt;/livro&gt;     &lt;/item&gt;   &lt;/itens&gt;   &lt;pagamento&gt;     &lt;status&gt;CONFIRMADO&lt;/status&gt;     &lt;valor&gt;29.90&lt;/valor&gt;     &lt;titular&gt;Edgar Bento&lt;/titular&gt;     &lt;email-titular&gt;edgar.b@abc.com&lt;/email-titular&gt;   &lt;/pagamento&gt; &lt;/pedido&gt;</pre>	

Depois, teremos uma mensagem com status *Enqueued*. Isso significa que a mensagem está pronta para ser entregue, ou pronta para ser consumida pelo Apache Camel.

Queues						
Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
pedidos	1	0	1	0	Browse Active Consumers Active Producers atom rss pdf	Send To Purge Delete

## Consumindo mensagens pelo Apache Camel

O Camel é integrado muito bem com JMS e ActiveMQ. Se for necessário acessar um broker JMS que não é ActiveMQ, existe um componente JMS:

<http://camel.apache.org/jms.html> (<http://camel.apache.org/jms.html>)

Como o ActiveMQ também é da Apache, foi criado um componente específico para ele:

<http://camel.apache.org/activemq.html> (<http://camel.apache.org/activemq.html>)

O primeiro passo é configurar o componente a partir do `CamelContext`. É preciso informar qual é o IP e a porta do broker, que é feito pelo método `addComponent`:

```
context.addComponent("activemq", ActiveMQComponent.activeMQComponent("tcp://localhost:61616"));
```

No segundo parâmetro adicionamos programaticamente o componente `ActiveMQComponent` ao Camel e o primeiro parâmetro será o apelido que usaremos no Camel DSL.

O componente `activemq` funciona com a mesma sintaxe dos outros componentes. Usaremos `:` e configurar se quisermos usar fila ou tópico seguido de "dois pontos" novamente para definir o nome do destino. Depois, vamos ler os pedidos da queue com o nome `pedidos`:

```
//usamos o componente activemq, consumindo da fila pedidos
from("activemq:queue:pedidos").
log("${file:name}").
routeId("rota-pedidos").
delay(1000).
to("validator:pedido.xsd").
log("chegamos aqui").
multicast().
    to("direct:soap").
        log("Chamando soap com ${body}").
    to("direct:http");
```

Conseguiremos consumir as mensagens! É incrível como o Camel consegue simplificar o código. Verificando a interface de administração do Apache Camel perceberemos que a mensagem foi entregue (*dequeued*):

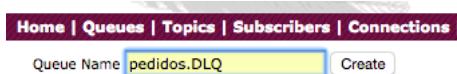
Queues						
Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
pedidos	0	1	1	1	Browse Active Consumers Active Producers atom rss pdf	Send To Purge Delete

## Usando DLQ no deadLetterChannel

No último capítulo, discutimos como o Camel lida com exceções e vimos como configurar o `deadLetterChannel`. Neste `deadLetterChannel`, usaremos uma fila JMS para guardar as mensagens venenosas, algo comum na integração. O JMS tem a vantagem de guardar as mensagens de maneira persistente e segura. Então, aproveitaremos essas características para combinar o `deadLetterChannel` com uma *Dead Letter Queue* (DLQ).

O primeiro passo é criar essa fila pela interface de administração do ActiveMQ: <http://localhost:8161/admin/queues.jsp> (<http://localhost:8161/admin/queues.jsp>).

Vamos chamar essa fila de `pedidos.DLQ`:



Em seguida, ajustar o `deadLetterChannel` para enviar as mensagens venenosas:

```
errorHandler(
    deadLetterChannel("activemq:queue:pedidos.DLQ"). //usando DLQ
        logExhaustedMessageHistory(true).
        maximumRedeliveries(3).
        redeliveryDelay(5000).
        onRedelivery(new Processor() {

            @Override
            public void process(Exchange exchange) throws Exception {
                int counter = (int) exchange.getIn().getHeader(Exchange.REDELIVERY_COUNTER);
                int max = (int) exchange.getIn().getHeader(Exchange.REDELIVERY_MAX_COUNTER);
                System.out.println("Redelivery - " + counter + "/" + max );
            }
        })
    );
}
```

Vamos pegar um XML de pedido que não passe pela validação e enviar pela interface do ActiveMQ para a fila `pedidos`. Após as tentativas de entrega o Camel deve entregar a mensagem para a DLQ. Para simplificar e focar apenas no DLQ, iremos ler os arquivos XML da pasta `pedidos`. Fica a seu critério, segue o código:

```
//from("activemq:queue:pedidos").
from("file:pedidos?delay=5s&noop=true").
log("${file:name}").
routeId("rota-pedidos").
//resto do código omitido
```

Não esqueça de rodar o Tomcat para inicializar os serviços web. Como resultado, o Camel deve entregar a XML do pedido que não passou pela validação a fila `pedidos.DLQ`.

Você pode acessar a DLQ pela interface do ActiveMQ:

<http://localhost:8161/admin/browse.jsp?JMSDestination=pedidos.DLQ> (<http://localhost:8161/admin/browse.jsp?JMSDestination=pedidos.DLQ>).

Veremos as mensagens (link *Browse* na coluna *Views*), deve aparecer o nosso XML do pedido que gerou a erro de validação:

The screenshot shows the Apache Camel message details interface. It includes sections for Headers, Properties, Message Actions, and Message Details.

Headers	Properties
Message ID ID:MacBook-Pro-de-Nico.local-56563-1448972745661- 1:1:1:1	CamelFileNameOnly 4_pedido.xml
Destination queue://pedidos.DLQ	CamelFileLastModified 1446656477000
Correlation ID	CamelFileRelativePath 4_pedido.xml
Group	CamelFileAbsolutePath /Users/nico/Documents/dev/workspaces/camel/camel-xml-pedido-temp/pedidos/4_pedido.xml
Sequence 0	CamelFileLength 806
Expiration 0	CamelFileName 4_pedido.xml
Persistence Persistent	CamelFilePath pedidos/4_pedido.xml
Priority 4	CamelFileParent pedidos
Redelivered false	breadcrumbId ID-MacBook-Pro-de-Nico-local-56557-1448972723314-0-35
Reply To	CamelFileNameConsumed 4_pedido.xml
Timestamp 2015-12-01 10:25:45:842 BRST	CamelFileAbsolute false
Type	

**Message Actions:** Delete, Copy, Move. The "Move" button has a tooltip: "... Please select ...".

**Message Details:**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<pedido>
<id>451256</id>
<dataCompra>2013-12-05T18:21:07.529-02:00</dataCompra>
<itens>
<item>
<formato>EBOOK</formato>
<quantidade>1</quantidade>
<livro>
```

## Para saber mais: Dependências do Maven

Nesse capítulo, usamos as duas dependências do Maven:

```
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-camel</artifactId>
    <version>5.6.0</version>
</dependency>

<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-jms</artifactId>
    <version>${camel-version}</version>
</dependency>
```

Segue o código das rotas:

```
errorHandler(
    deadLetterChannel("activemq:queue:pedidos.DLQ").
//      logExhaustedMessageHistory(true).
      maximumRedeliveries(3).
      redeliveryDelay(5000).
    onRedelivery(new Processor() {

        @Override
        public void process(Exchange exchange) throws Exception {
            int counter = (int) exchange.getIn().getHeader(Exchange.REDELIVERY_COUNTER).
            int max = (int) exchange.getIn().getHeader(Exchange.REDELIVERY_MAX_COUNTER).
            System.out.println("Redelivery - " + counter + "/" + max );
        }
    })
```

```

        })
;

//from("activemq:queue:pedidos").
from("activemq:queue:pedidos")
    log("${file:name}").
    routeId("rota-pedidos").
    delay(1000).
    to("validator:pedido.xsd").
    multicast().
        to("direct:soap").
        //log("Chamando soap com ${body}").
        to("direct:http");

from("direct:soap").
    routeId("rota-soap").
to("xslt:pedido-para-soap.xslt").
    //log("Resultado do template: ${body}").
    setHeader(Exchange.CONTENT_TYPE,constant("text/xml")).
to("http4://localhost:8080/webservices/financeiro");

from("direct:http").
    routeId("rota-http").
        setProperty("pedidoId", xpath("/pedido/id/text()")).
        setProperty("email", xpath("/pedido/pagamento/email-titular/text()")).
        split().
            xpath("/pedido/itens/item").
        filter().
            xpath("/item/formato[text()='EBOOK']").
        setHeader("CamelFileName", simple("${file:name}.json")).
        setProperty("ebookId", xpath("/item/livro/codigo/text()")).
        setHeader(Exchange.HTTP_QUERY,
            simple("clientId=${property.email}&pedidoId=${property.pedidoId}&ebookId=${pro
to("http4://localhost:8080/webservices/ebook/item");
}

```

## O que aprendemos?

- Cadastrar um componente programaticamente;
- Consumir mensagens JMS com Camel;
- Enviar mensagens JMS;
- Usando um DLQ para o deadLetterChannel.

[1]: [\(https://s3.amazonaws.com/caelum-online-public/camel/camel-stage-cap7.zip\)](https://s3.amazonaws.com/caelum-online-public/camel/camel-stage-cap7.zip)